From Higher-Order $\pi$-Calculus to $\pi$-Calculus in the Presence of Static Operators

José-Luis Vivas$^1*$ and Mads Dam$^2**$

$^1$ Dept. of Teleinformatics, Royal Institute of Technology, Stockholm
$^2$ SICS, Swedish Institute of Computer Science, Stockholm

Abstract. Some applications of higher-order processes require better control of communication capabilities than what is provided by the $\pi$-calculus primitives. In particular we have found the dynamic restriction operator of CHOCS, here called blocking, useful. We investigate the consequences of adding static operators such as blocking to the first- and higher-order $\pi$-calculus. In the presence of the blocking operator (and static operators in general) the higher-order reduction of Sangiorgi, used to demonstrate the reducibility of higher-order communication features to first-order ones, breaks down. We show, as our main result, that the higher-order reduction can be regained, using an approach by which higher-order communications are replaced, roughly, by the transmission and dynamic interpretation of syntax trees. However, the reduction is very indirect, and not usable in practice. This throws new light on the position that higher-order features in the $\pi$-calculus are superfluous and not needed in practice.

1 Introduction

One of the most significant contributions of the $\pi$-calculus has been the demonstration that higher-order features in concurrency can be eliminated in favour of first-order ones by means of channel name generation and communication. This issue has been extensively studied in the context of lambda-calculus under various evaluation regimes (cf. [3]), and in his thesis [8] Sangiorgi explored in depth the reduction of higher-order processes to first-order ones. Instead of communicating a higher-order object, a local copy is created, protected by a trigger in the shape of a newly generated channel name. This trigger can then be communicated in place of the higher-order object itself. On the basis of this sort of reduction it has been argued (cf. [8]) that, in the context of the $\pi$-calculus, higher-order features are matters of convenience only: No essential descriptive or analytical power is added by the higher-order features.

In this paper we reexamine this position, and find it borne out in principle, but not in practice. We argue the following points:

* Supported by the Swedish National Board for Technical and Industrial Development (NUTEK) under grant no. 94-06164. Email: josev@it.kth.se
** Supported by a Swedish Foundation for Strategic Research Junior Individual Grant. Email: mfd@sics.se
1. Practical applications call for process combinators other than those provided by the basic \( \pi \)-calculus. Specifically we consider the dynamic restriction, or blocking\(^1\) operator of Thomsen’s CHOCS \([9]\).
2. Adding blocking to the higher-order \( \pi \)-calculus causes Sangiorgi’s reduction to break down.
3. In the presence of blocking it remains possible to reduce the higher-order calculus to the first-order one, even in a compositional manner.
4. The reduction, however, is complicated, and amounts in effect to the communication and interpretation of parse trees. In contrast to Sangiorgi’s reduction which is conceptually quite simple this reduction can not be used in practice to reduce non-trivial arguments concerning higher-order processes to arguments concerning first-order ones.

Our reduction is very general and can be applied to a wide range of static process combinators. Our specific interest in the blocking operator stems from some difficulties connected with the representation of cryptographic protocols in the higher-order \( \pi \)-calculus \([2]\).  

**Application: Cryptographic Protocols** Consider a higher-order process of the shape \( A = \text{\texttt{\#m}}.A \). The process \( A \) is an object which repeatedly outputs \( m \) along \( a \) to whomever possesses knowledge of \( a \) and is willing to listen. In principle \( m \) can be any sort of higher-order object, but here it suffices to think of \( m \) as a message carried by \( a \). A cryptographic analogy of \( A \) is thus the object \( \{m\}_a, \) \( m \) encrypted by the (shared) encryption key \( a \). We might very well want to communicate \( A \) as a higher-order object over some other, possibly insecure, channel \texttt{xfer}. The sender would simply pass \( A \) along \texttt{xfer}, and the receiver would first receive some process object \( X \), then immediately activate \( X \) while in parallel trying to extract the message \( m \) through \( a \). That is, the receiver would have the shape \( \texttt{xfer}(X).(X | a(y).B(y)) \) where \( B(y) \) is the continuation processing the extracted \( y \) in some suitable way. Observe that we can assume receivers and senders to execute in an environment containing other receivers and senders, along with unknown and possibly hostile intruders.

Here we encounter a first difficulty: We have provided no guarantee that it is really \( X \) and \( B(y) \) which communicate along \( a \) and not some other process which is trying to decrypt using \( a \) by accident or because of some protocol flaw. That is, decryption is insecure, contradicting commonly held assumptions in the analysis of key management protocols. When encryption is nested, however, the problem is aggravated. A higher-order representation of \( \{\{m\}_{a}\}_b \) is the object \( A' = bA.A \). Extraction of \( m \) from \( A' \) would follow the pattern

\[
\texttt{xfer}(X).(X | b(Y).(Y | a(z).B(z))).
\]

\(^1\)Since it is not completely clear in which senses the restriction operators are really static or dynamic we prefer a more neutral terminology and use “restriction” for the \( \pi \)-calculus restriction operator and “blocking” for the CHOCS dynamic restriction operator.